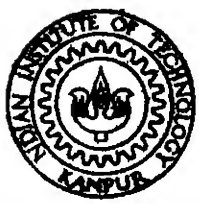
MICROPROCESSOR BASED SPECTRAL ANALYSIS FOR REAL TIME APPLICATIONS

By

S V SESHAGIRI RAO



DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

AUGUST 1980

MICROPROCESSOR BASED SPECTRAL ANALYSIS FOR REAL TIME APPLICATIONS

A Th i
S bmitt d i P i l Fulfim t f th Req i m t
f th D g f
MASTER OF TECHNOLOGY

By

S V SESHAGIRI RAO

to the

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

AUGUST 1980

CERTIFICATE

This is to certify that the work entitled 'MICROPROCESSOR BASED SPECTRAL ANALYSIS FOR REAL-TIME APPLICATIONS' has been carried out by Mr S V Seshagiri Rao under my supervision and this has not been submitted elsewhere for a degree

August, 1980

(DR S K MULLICK)
PROFESSOR
Department of Electrical Engineering
Indian Institute of Technology
KANPUR

ACKNOWLEDGEMENT

I am deeply indebted to Prof S K Mullick for suggesting this problem and also for his constant encouragement and guidance throughout the course of the work

I am very thankful to Prof V P Sinha, Dr M U Siddiqi, for sparing their valuable time for the useful discussions I had with them, during this work

I thank Mr B V Ramana, Research Engineer, A C E S, for his guidance in microprocessor Lab I al o thank Lt S R Muthangi, F Lt S Visweswaraih, Mr BRK Reddy and all of my friends who have directly or indirectly helped me during this work

I finally thank, Mr Y Chandra, for the excellent typing work

August, 1980 S V, Seshagiri Rao

CONTENTS

A B S T R A C T

With the advent of microprocessors, the DFT computation for spectral analysis has become very easy and economical, But to carry out spectral analysis for real-time applications the choice of appropriate algorithm is imminent 'Recursive way of computing DFT' is the most suitable one for the said purpose. After briefly reviewing the various algorithms for computing DFT, this work discusses the implementation of a known recursive algortihm on an Intel 8080 microprocessor based computer (ECIL'S MICRO-78) The advantages of this algorithm when compared with FFT algorithm are clearly brought out in this work by making a comparative study of both the methods Several examples are considered to demonstrate the utility of the algorithm in computing the DFT recursively A specific mention is made about the inability of the developed assmebly language program for carrying out spectral analysis of high frequency signals because of the short comings of the microprocessor used An economical and highly reliable real-time spectrum analyzer scheme is proposed for actual implementation In conclusion an outline for further work based on the concept of recursive algorithm is also suggested

CHAPTER 1

INTRODUCTION

In recent years digital signal processing techniques have gained great importance in many areas of science and technology This is mainly due to the fact of increasing availability of medium and large scale digital integrated circuits with their desirable properties of small size, low power requirement, low cost, noise immunity and high reliability

'Spectral analysis' of discrete time signals is a very important area of application of the digital signal processing Spectral analysis, in general refers to the various alternative descriptions of a signal in terms of a linear combination of chosen basis set of representation Fourier series and Fourier integral forms of representation are of fundamental importance because the exponential functions are eigen functions of a Linear time in-variant (LTI) systems Spectral analysis has got many diversified applications The major areas in which spectral analysis is extensively used are Electronics and Communications Engineering, Mechanical Vibrations and Structural analysis and in general for time series analysis in all endeavours of scientific and technological activity to detect, identify and characterize the natural and man made processes

which change in time or space

For example a high fidelity amplifier should not distort signals in the range between 50Hz and 20K Hz Spectral analysis is useful for checking this Another example is the use of sound spectra to identify the person whose spoken voice has been recorded This is a speaker identification problem which is mainly useful for crime detection Transmission system's properties can also be described in the frequency domain so that the system can be designed for undistorted transmission of the frequencies present in the actual signal that is being transmitted Spectral analysis is used for detection of signals burried in noise In air traffic problems the frequency content of a plane's trajectory corresponds to the sharpness of bends in the plot of this trajectory on a radar screen The range of possible frequencies in this curve will determine how close together in time we must take position samples for a digital signal processor which predicts the plane's future position A radio Astronomer may be able to deduce information in a planet's atmosphere on the basis of relative strengths of the planet's radiation in various frequency bands Mass spectrography and crystalographic analysis are among the many areas in which frequency analysis is used

Even for large amount of data, spectral analysis has become a considerably easier affair with the advent of digital signal

processing either by digital computer or dedicated hardware Further it has become very economic with the advent of LSI technology and microprocessors The use of such devices for the purpose of signal processing necessitate the description of the analog signal from various transducers in the discrete and digital domain

Thus the concept of Fourier representation of analog signals is extended to data sequences and digital signals The result is 'Discrete Fourier Transform' (DFT) This form is nothing but discrete-time and discrete-frequency representation of Fourier transform

With the advent of small, inexpensive micro-computers and with the developed algorithms for efficiently computing DFT, such as 'fast Fourier Transform'(FFT), Winograd Fourier Transform (WFT), Fourier Transform computers using coordinate rotation digital computer (CORDIC) iterations and recursive ways of computing DFT, the utility and applications of DFT, has increased enourmously The applications include analysis of mechanical and structural vibrations, the extraction of vital information from radar and sonar signals, analysis of statistical data, digital filtering and many others

To carry out the spectral analysis in real time we have to compute the DFT as the data come and without any storage After

a survey of algorithms available to compute DFT in an efficient manner along with the constraint of 'real time spectral analysis', 'recursive way of computing DFT' is found the most suitable So in this present work recursive algorithm is made use of for finding out the DFT of a set of sample points This type of spectral analysis is useful especially when the data are coming sequentially and when there is sufficient time gap between samples This algorithm has certain advantages as compared to computation of DFT using FFT algorithms A comparative study has been conducted and the results are presented in later Chapters

The recursive algorithm for computing DFT has been implemented on a micro-computer(ECIL's MICRO-78) which is built around Intel 8080 microprocessor This particular machine is selected mainly because of the availability of this microprocessor with a number of required interface, peripheral devices and support software

The DFT for different sizes of data, obtained from time sampling of a representative set of signals has been calculated on the micro-computer with the help of a recursive algorithm An implementation of the same algorithm on DEC-10 computer yielded approximately the same results

Some methods or algorithms available to compute DFT of a given set of data points and the theoretical formulation of

the recursive algorithm will be dealt in Chapter 2 The details of microprocessor implementation of the recursive algorithm are presented in Chapter 3 Chapter 4 is devoted to illustrative examples of DFT calculation using the recursive algorithm Here as a particular application, DFT is utilized in the computation of power spectrum estimate which is a very useful characterization of stationary random signals

Chapter 5 undertakes a comparative study of computational time and memory requirement aspects in the computation of DFT using a) recursive algorithm b) FFT A brief report on the conclusions of the work and the scop of future work is mentioned in the last Chapter

CHAPTER 2

THEORETICAL FORMULATIONS

A number of algorithms are available to efficiently compute DFT of a set of data points We have to select one particular algorithm depending on many factors such as 1) the size of data for which DFT is computed 2) the particular application in mind 3) the speed of the processor at hand which computes the DFT. The following algorithms are available in literature for computing DFT

1) DFT by use of FFT algorithms
2) The DFT algorithm of Winograd (WFT)
3) DFT using CORDIC (Coordinate Rotation Digital Computer), VFFT (very Fast Fourier Transform)
4) Recursive algorithm for DFT
5) State variable approach in calculating DFT (Recursive algorithm)

In succeeding sections of this Chapter a brief mention of the various algorithms is made and finally the details of Recursive algorithm are given in Section (2 8).

2 1 THE FAST FOURIER TRANSFORM (FFT)

The FFT is simply an algorithm (a particular method of performing a series of computations) that can compute the DFT

much more rapidly than the direct evaluation of DFT

DFT is defined as

$$X_j = \sum_{k=0}^{M-1} Z_k \quad (W)^{jk} \qquad (2.1)$$

$$j=0,1, \qquad M-1$$

where

$W = \exp(-2\pi i/M)$

$i = \sqrt{-1}$

The computation of DFT can be written in matrix form as follows (for $M = 4$)

$$\begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 \\ W^0 & W^2 & W^4 & W^6 \\ W^0 & W^3 & W^6 & W^9 \end{bmatrix} \begin{bmatrix} Z_0 \\ Z_1 \\ Z_2 \\ Z_3 \end{bmatrix} \qquad (2\ 2)$$

or more compactly

$$X(n) = W^{jk} \quad Z(k)$$

If samples are complex numbers we need M^2 complex multiplications and $M(M-1)$ complex additions By suitably factorizing W matrix we reduce the multiplications The details may be found in Reference (1) Essentially for $M=2^{\gamma}$, the FFT algorithm is a procedure for factoring MXM matrix into γ matrices each

(MXM) such that each of the factored matrices has the special property of minimizing the number of complex multiplications and additions FFT requires $M\gamma/2$ complex multiplications and $M\gamma$ complex addition, where as the direct method requires M^2 complex multiplications & $M(M-1)$ complex additions Thus large saving will be there in the computational time The above is one method of computing DFT efficiently

2 2 FFT ALGORITHM DETAILS

In the following section the fast Fourier transform (FFT) algorithm is briefly described This is considered worth describing because this brings out clearly some of the advantages in DFT computation by recursive way of computation

Considering the DFT relationship

$$X(n) = \sum_{k=0}^{N-1} X_0(k)\, W^{nk} \qquad (2\ 4)$$

$$n = 0,1, \qquad N-1$$

$(N = 2^{\gamma})$ where $W = e^{-j\,2\pi/N}$

For $N = 4$, $\gamma = 2$ and k and n are represented 2 bit binary numbers

$$k = 0,\ 1,\ 2,\ 3 \quad \text{or} \quad k = (k_1\ ,\ k_0)$$

$$= 00,\ 01,\ 10,\ 11$$

$$n = 0, 1, 2, 3 \quad \text{or} \quad n = (n_1, n_0)$$
$$= 00, 01, 10, 11$$

A compact method of writing k and n is $k = 2k_1 + k_0$, $n = 2n_1 + n_0$ where k_0, k_1, and n_1 can take on values of 0 and 1 only

Using the above representation for N=4 the DFT can be written as

$$X(n_1,n_0) = \sum_{k_0=0}^{1} \sum_{k_1=0}^{1} X_0(k_1, k_0) W^{(2n_1+n_0)(2K_1+K_0)} \qquad (2\ 5)$$

It should be observed that single summation in DFT original definition, equation (2 4) is now replaced by Y (=2) summations in equation (2 5), in order to enumerate all the bits of the binary representation of k

Factorization of w^P

$$W^{(2n_1 + n_0)(2k_1 + k_0)} = W^{(2n_1 + n_0) 2k_1} \; W^{(2n_1 + n_0)k_0}$$

$$= (W^{4n_1k_1}) \quad (W^{2n_0k_1}) \quad (W^{(2n_1 + n_0) k_0})$$

$$= (W^{2n_0k_1}) \; (W^{(2n_1 + n_0) k_0})$$

This is because, $W^{4n_1k_1} = (W^4)^{n_1k_1}$

$$\left(e^{\frac{-j\,2\pi x\,4}{4}}\right)^{n_1k_1} = (1)^{n_1k_1} = 1$$

Then the equation (2 5) can be written in the form

$$X(n_1,n_0) = \sum_{k_0=0}^{1} \left[\sum_{k_1=0}^{1} x_0(k_1,k_0)\, W^{2n_0k_1} \right] W^{(2n_1+n_0)k_0} \qquad (2\ 6)$$

Equation (2 6) represents the foundation of the FFT algorithm To demonstrate this point, let us consider each of the summations of equation (2 6) individually

If the summation in the brackets is rewritten

$$x_1(n_0,k_0) = \sum_{k_1=0}^{1} x_0(k_1,k_0)\, W^{2n_0k_1} \qquad (2\ 7)$$

$$x_1(0,\ 0) = x_0(0,\ 0) + x_0(1,\ 0)\ W^0$$
$$x_1(0,\ 1) = x_0(0,\ 1) + x_0(1,\ 1)\ W^0$$
$$x_1(1,\ 1) = x_0(0,\ 0) + x_0(1,\ 0)\ W^2$$
$$x_1(1,\ 1) = x_0(0,\ 1) + x_0(1,\ 1)\ W^2$$

If those equations are written in matrix form as

$$\begin{bmatrix} x_1(0,0) \\ x_1(0,1) \\ x_1(1,0) \\ x_1(1,1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & W^0 & 0 \\ 0 & 1 & 0 & W^0 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^2 \end{bmatrix} \begin{bmatrix} x_0(0,0) \\ x_0(0,1) \\ x_0(1,0) \\ x_0(1,1) \end{bmatrix} \quad (2\ 8)$$

Thus, the inner summation of equation (2 6) specifies the first of the factored matrices for the example

Similarly, if the outer summation of equation (2 6) is written as

$$X_2(n_0, n_1) = \sum_{k_0=0}^{1} X_1(n_0, k_0)\, W^{(2n_1+n_0)\,k_0} \quad (2\ 9)$$

and can be written in matrix form as :

$$\begin{bmatrix} x_2(0,0) \\ x_2(0,1) \\ x_2(1,0) \\ x_2(1,1) \end{bmatrix} = \begin{bmatrix} 1 & W^0 & 0 & 0 \\ 1 & W^2 & 0 & 0 \\ 0 & 0 & 1 & W^1 \\ 0 & 0 & 1 & W^3 \end{bmatrix} \begin{bmatrix} x_1(0,0) \\ x_1(0,1) \\ x_1(1,0) \\ x_1(1,1) \end{bmatrix} \quad (2\ 10)$$

Thus the outer summation results in the second of the factored matrix

From equations (2,6), (2 9) one can observe that

$$X(n_1, n_0) = x_2(n_0, n_1) \quad (2\ 11)$$

If the equations (2 7), (2 9), (2 11) are combined the following set of equations will be obtained

$$x_1(n_0, k_0) = \sum_{k_1=0}^{1} x_0(k_1, k_0)\ W^{2n_0k_1}$$

$$x_2(n_0, k_1) = \sum_{k_0=0}^{1} x_1(n_0, k_0)\ W^{(2n_1 + n_0)k_0} \qquad (2\ 12)$$

$$X(n_1, n_0) = x_2(n_0, n_1)$$

This set of equations (2 12) represent the original Cooley - Tukey algorithm of FFT

The above equations are represented by the signal flow-graph and is shown in Fig 2 1

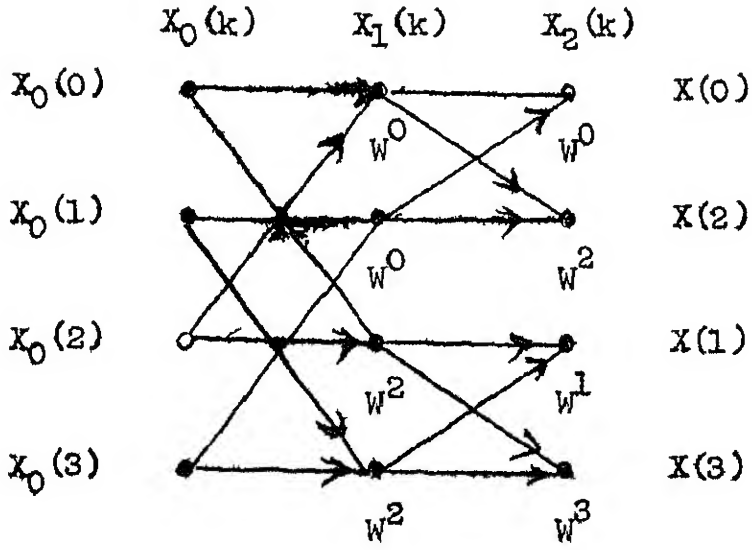


Fig. 2.1

It is observed that input data is in natural order and output is in scrambled form One has to interchange nodes (01) and (10) in each array to get output in natural order and input data then becomes scrambled one In FFT algorithm there should be one sub-routine which calculates the bit reversal of a numbers (unscrambling) This difficulty is not there in recursive algorithm to be dealt in later sections After scrambling the input data the signal graph will look like as shown in (Fig 2 2)

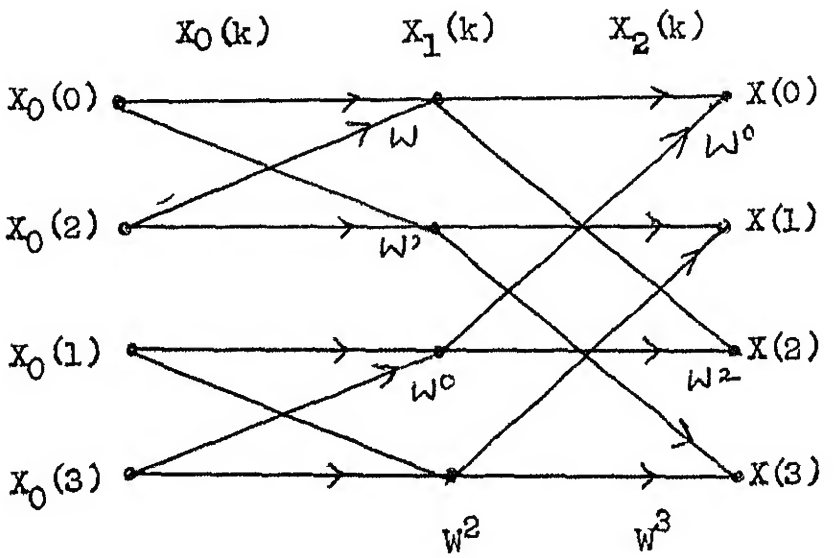


Fig 2 2

The two forms of the algorithms are referred by the term decimation in time The terminology arises from the fact that

alternate derivations of the algorithm are structured to appeal to the concept of sample rate reduction of throwing away samples and thus the term decimation in time

Sande-Tukey developed similar a FFT algorithm In contrast to Cooley-Tukey approach here components of n are seperated instead of the components of k Finally here the following equations are obtained

$$x_1(n_0, k_0) = \sum_{k_1=0}^{1} x_0(k_1, k_0)\, W^{2n_0k_1}\, W^{n_0k_0}$$

$$x_2(n_0, n_1) = \sum_{k_0=0}^{1} x_1(n_0, k_0)\, W^{2n_1,k_0}$$

$$X(n_1,n_0) = x_2(n_0,n_1)$$

This form of the algorithm is known by the term decimation in frequency and the reasoning for the terminology is analogous to that for decimation in time Other details can be found in Ref (1).

2 3 FOURIER TRANSFORM COMPUTERS USING CORDIC ITERATIONS

The basic Discrete Fourier Transform (DFT) is defined as

$$W_j = \sum_{K=0}^{M-1} Z_K \quad (W)^{JK} \qquad (2\ 14)$$

$$j = 0,\ 1, \qquad M-1$$

where

$$W = \exp\ (-j2\pi i/M)$$

$$i = (-1)^{\frac{1}{2}}$$

Z_0, Z_1 Z_{M-1} are samples of the waveform

In this method of computing DFT the transform may be viewed as a process of rotating the vector Z_K (real and imaginary components) by the angle $(2\pi jK/M)$, followed by a sum over all K for each j The CORDIC vector Rotation method of Volder Ref (2), is used for the computation of DFT This appears to be attractive as a vector can be rotated in the time of a single multiply using this technique

In addition, trigonometric coefficients as distinct from the CORDIC constants are not needed in a CORDIC rotation procedure so the expense of generating, storing and retrieving these coefficients is avoided

2 4 CORDIC TECHNIQUE

Consider the vector $X_i = (x,y)$ to be rotated by angle $= 2\pi jK/M$ Let a new vector $\wedge_{i+1}$ be obtained from X_i by the process

$$\begin{aligned} x_{i+1} &= x_i + y_i \ 2^{-i} \\ y_{i+1} &= y_i - x_i \ 2^{-i} \end{aligned} \qquad (2\ 15)$$

Then if

$$\begin{aligned} R_i &= x_i^2 + y_i^2 \\ \theta_i &= \mathrm{Tan}^{-1}(y_i/x_i) \end{aligned} \qquad (2\ 16)$$

the result will be that the magnitude and the angle of the new vector

$$X_{i+1} = (x_{i+1} , y_{i+1}) \qquad (2\ 17)$$

will be

$$R_{i+1} = R_i[1+2^{-2i}]^{\frac{1}{2}}$$

$$\theta_{i+1} = \theta_i - \tan^{-1}(2^{-i})$$

Now to rotate X by an arbitrary angle α between $(\pm \pi/2)$ to an accuracy of 1 bit in n bits, the following procedure

may be used

Initialize $i = 0$

$$Z_i = -\alpha$$

Iterate n times

$$\alpha_i = \mathrm{Sgn}(Z_i)$$
$$x_{i+1} = x_i + \alpha_i y_i 2^{-i}$$
$$y_{i+1} = y_i - \alpha_i x_i 2^{-i}$$
$$Z_{i+1} = Z_i - \alpha_i \tan^{-1}(2^{-i})$$
$$i = i + 1$$

These iterations are composed of addition, subtraction, shifting (multiplication by 2^{-i}) and table look up ($\tan^{-1}(2^{-i})$)

2 5 APPLICATION TO DFT

The application of CORDIC technique to DFT is as follows

One rotation will be required for each combination of input and output for N^2 total revolutions associated with N^2 multiplications in the computation of DFT The real and imaginary components of the input vector Z_k are taken as the initial values of x_i and y_i. The initial value of Z_i is then set to $2\pi jk/M$ The CORDIC iteration is performed n times to produce n bits of precision and th resulting x and y

values are added to the sum that will become W_j after M repetitions of this process The process is repeated for each frequency component W_j We also note that the resulting spectrum is scaled by a constant factor k where

$$k = \prod_{i=0}^{n-1} (1 + 2^{-2i})^{\frac{1}{2}} \qquad 1\ 6 \qquad (2\ 19)$$

A very low-cost Fourier transform computer can be constructed to perform DFT as described above It will be most useful for those cases of modest input data rate, where only a few high resolution fr quency samples are required for the output In this case considerable storage of trigonometric coefficients , unnecessary frequency coefficients and buffers can be saved as compared with the FFT approach

The basic idea of application of CORDIC technique to DFT computation has recently been reviewed by ALVIN M DESPAIN, Ref (3) who has proved that by suitable modification of CORDIC, DFT computation will involve no multiplication The following description gives an idea of DESPAIN's very Fast Fourier Transform (VFFT) using CORDIC techniques

2 6 VERY FAST FOURIER TRANSFORM ALGORITHM

This algorithm provides procedure for computation of DFT that involves no multiplications As DFT can be viewed

as a sum of rotated complex vectors, CORDIC techniques can be employed to accomplish the rotations It was also shown in the previous section (2 4) that an overall constant change of magnitude of the rotated vector does not disturb the fundamental nature of the DFT but acts as a constant gain The acceptance of an arbitrary gain factor simplified the CORDIC algorithms considerably In this particular algorithm (VFFT) it is visualized that all the angles can also be rotated by a fixed, constant angle Then the gain constant is complex and, if necessary, can be compensated for, in a final correction after the total DFT is accomplished As a result, multiplications can be eliminated and the number of arithmetic operations required in the calculation of DFT can be considerably fewer than by other methods This results in considerable saving in hardware Because the algorithms require no multiplications and require only N storage locations (the so called 'inplace" property) they are of great advantage in microprocessor programs for signal processing tasks The details of the algorithm can be found in Ref (3)

In the following paragraphs very fast Fourier transform (VFFT) algorithm is briefly described

DFT is defined as

$$A_r = \sum_{k=0}^{N-1} B_K \quad W_N^{rk} \qquad r = 0,1,2 \qquad N-1 \qquad (2\ 20)$$

where

$$W_N = \exp(-2\pi j/N)$$

$$j = \sqrt{-1}$$

Here it is assumed that N has at least one factor equal to 16 and the DFT should be calculated with as little effort as possible The radix-16 algorithm can be easily derived from equation (2 20) Given $N = 16T$, let $r = s + Tt$ and $K = 1 + 16m$

Now eqn (2 20) becomes

$$A_{s+tT} = \sum_{l=0}^{15} \sum_{m=0}^{T-1} B_{1+16m} W_{16T}^{(s+Tt)(1+16m)}$$

$$= \sum_{l=0}^{15} \sum_{m=0}^{T-1} B_{1+16m} W_{16T}^{sl} W_{16}^{Ttl} W_{16T}^{16Sm} W_{16T}^{16mtT}$$

but

$$W_{16T}^{mtT} = 1$$

and $$W_{16T}^{16ltT} = W_{16}^{lt}$$

$$W_{16T}^{16Sm} = W_T^{Sm}$$

Thus

$$A_{S+tT} = \sum_{l=0}^{15} W_{16}^{lt} \; W_{16T}^{Sl} \sum_{m=0}^{T-1} B_{l+16m} \; W_T^{Sm} \qquad (2\ 21)$$

Now let

$$B'_{S,l} = W_{16T}^{Sl} \sum_{m=0}^{T-1} B_{l+16m} \; W_T^{Sm} \qquad (2\ 22)$$

and (2 21) becomes

$$A_{S+tT} = \sum_{l=0}^{15} B'_{Sl} \; W_{16}^{lt} \qquad (2\ 23)$$

This equation (2 23) is the DFT for N = 16 while (2 22) is the DFT for N = T multiplied by the factor W_{16T}^{Sl} This procedure can be applied to eqn (2 22) until all possible 16-point DFT'S have been realized (i e T no longer has 16 as a factor)

The equation (2 23) can be manipulated and can be converted to radix-4 representation And finally the 4-point DFT is converted to 2 point DFT after repeating the same mathematical manipulation that yielded 4-point DFT from 16-point DFT Fig 2 3 is a block diagram illustrating 'recursive radix" representation of a 16-point DFT In the above derivation, the only term which involves more than simple addition or

subtraction is the vector rotation operator W_{16}^{i} It can be converted to a form in which only a minimal number of additons, subtractions, and shift (multiply by 2^{i}) operations occur

Determination of W_{16}^{i} :

Table I illustrates the vector rotations of W_{16}^{i} as a function of i The method used to calculate W_{16}^{i} is to employ rotations of the form

$$x' \leftarrow f_1(x) \mp f_2(y)$$

$$y' \leftarrow f_1(y) \pm f_2(x)$$

where $\vec{A} = x + \sqrt{-1}\, y$

and f_1 and f_2 are restricted so as to involve only additions, subtractions and multiplication by 2^{i}(shift t_s)

The resulting rotation θ is

$$\theta_i = \pm \tan^{-1} (\pm f_2( )/f_1( )) \qquad (2\ 24)$$

while the magnitude of A is multiplied by G :

$$G = \sqrt{f_1^2( ) + f_2^2 ( )}$$

For the CORDIC case $f_1( ) = ( )$, $f_2( ) = ( )2^{-i}$ and iterations for i = 0, 1, 2, n-1 are employed to rotate by a desired

angle with n bits of precision in the result This requires n additons and n subtractions Another approach is to implement the four rotations θ_1, θ_2, θ_3 and θ_4 shown in Table I and compensate later for the bias

Rotation by $\pm\theta$:

To obtain the rotation by $\pm\theta$ then form equation (2 24)

$$\pm\theta = \tan^{-1}(\beta/\alpha) \text{ or } \quad \beta/\alpha = \tan(\theta)$$

One has to choose and β to produce θ to the desired degree of precision, and such that the corresponding equations

$$x' \leftarrow f_1(x) \mp f_2(y)$$

$$y' \leftarrow f_1(y) \pm f_2(x)$$

contain only a minimum number of additions and subtractions Solutions for various levels of precision were generated by use of the following algorithm

Algorithm

"Find integers α β such that $\alpha/\beta = R$ to within $\pm \frac{1}{2}$ bit in N bits'

Begin $\alpha \leftarrow 0$

$\varepsilon \leftarrow 2^{N-1}$

while ($\alpha \leftarrow 2^{N-1}$)

```
Begin  α ← α + 1
       β ← Integer part (α R - ε)
while ( β < α R + ε + 1)
    Begin
        Error ← |( β/α ) - R|
    If (Error ≤ ε ): Display(α , β , Error)
                β ← β + 1
                End,
```

APPLICATION

A Radix - 16 Pipeline Cascade

To illustrate the application of the above principle in an FFT processor, a 16 point DFT cascade circuit with a precision of 1 bit in 16 bits (dynamic range = 90 dB) will be derived This circuit could be employed within processors that provide DFT of any size which includes 16 as a factor The basic form is shown in Fig 2 3

The 2-point DFT is of the form

$$A_r = \sum_{k=0}^{1} B_K \; W_2^{rK}$$

Thus $A_0 = B_0 + B_1$

and $A_1 = B_0 - B_1$

A digital circuit to implement these operations is shown in Fig 2 4

The rotation W_4 of Fig 2 1 can be expressed as

$$W_4^{uf} = (\exp(-2\pi i/4))^{uf}$$

$$= \exp\left(\frac{-\pi i}{2} uf\right)$$

Thus

$$W_4^{uf} = (-i)^{uf \bmod 4}$$

or $W_4^0 = 1$

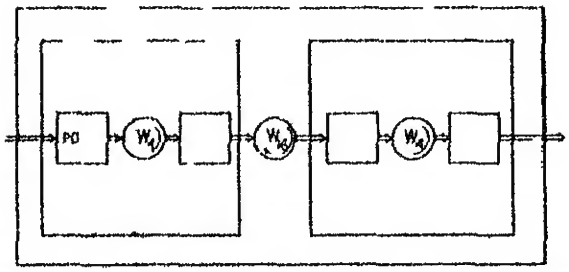
$W_4^1 = -i$

$W_4^2 = -1$

$W_4^3 = +i$

$W_4^4 = 1$

A digital circuit to implement this equation is shown in Fig 2 5



F'g2-3 Recu i adi p ese tl f 16-po t DFT cascade

TABLE I

R i ns f | (Bias 3 /16)					
i (Inde)	Ideal R ion (radi ns)	M m m Bias R prese on			
		Θ	Θ	Θ	Θ
0	0	0	0	/8	/16
1	/8	0	0	/8	+ /16
2	/4	0	0	+ /8	/16
3	3 /8	0	0	+ /8	+ /16
4	/2	0	/2	-π/8	/16
5	5 /8	0	/2	/8	+ /16
6	3 /4	0	/2	+ /8	/16
7	7π/8	0	/2	+ /8	+ /16
8			0	/8	/16
9	9 /8		0	/8	+ /16
10	5 /4		0	+ /8	/16
11	11 /8		0	+ /8	/16
12	3 /2		/2	/8	/16
13	13 /8		/2	/8	+ /16
14	7 /4		π/2	+ /8	/16
15	15 /8		/2	+π/8	+ /16

The last operation of Fig 2 1 is W_{16}^{uf}. Let i = uf; then one can use the representation of Table I for W_{16}^{i} The rotations θ_1, θ_2 are implemented with circuit of Fig 2 5 as explained above rotation by θ_3 may be implemented by reference to Table II(a) To obtain at least 16-bit precision, eight adders will be needed to realize the ratio

$$(f_2/f_1) = (128/309)_{\text{decimal}} =$$

$$(0\ 010000000\ /\ 0\ 0100110101)_{\text{binary}}$$

then x' and y' are given by

$$x' \leftarrow 0\ 01000110\ x \mp 0\ 10011010y)$$

$$y' \leftarrow 0\ 100110101y \pm 0\ 010000000x$$

or

$$x' = ((x + x2^{-2}) - (x + x\ 2^{-2})\ 2^{-5})\ x\ 2^{-8} \mp y\ 2^{-2}$$

$$y' = ((y + y2^{-2}) - (y + y\ 2^{-2})\ 2^{-5}) - 62^{-8} \pm x2^{-2}$$

and if we define $tx = x + x2^{-2}$

$$ty = y + y2^{-2}$$

then the equations become

$$x' = (tx - tx\ 2^{-5} - x\ 2^{-8})\ 2^{-1} \mp y2^{-2}$$

$$y' = (ty - ty2^{-5} - y2^{-8})\ 2^{-1} \pm x2^{-2}$$

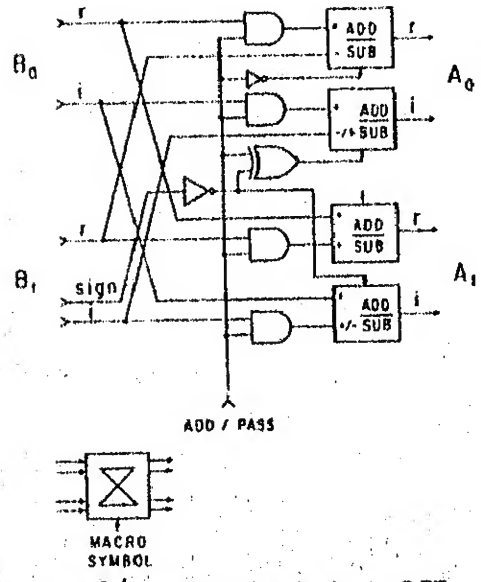


Fig 2.4 Digital circuit for a 2-point DFT.

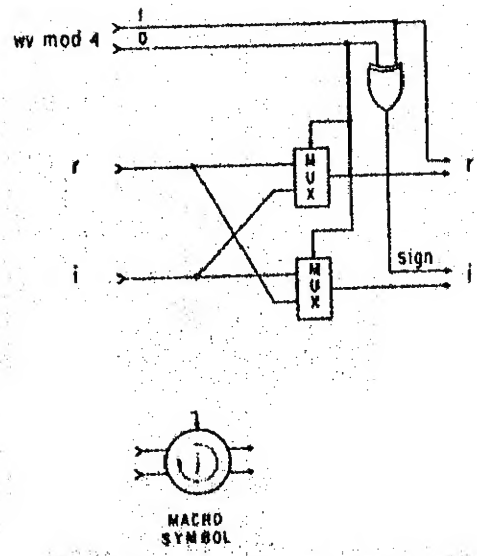


Fig 2.5 Digital circuit to implement $W_4^{\cdot}$.

and it is apparent that eight adders all needed Fig 2.6 illustrates the ckt rotation by θ_4 is accomplished in a similar manner using Table I & II (b)

Here $(f_2/f_1) = (\sqrt{5})((-2^{-10}))$

$$x^1 \leftarrow (x + x\,2^{-2}) \mp y\,2^{-2}$$
$$y^1 \leftarrow (y + y\,2^{-2}) \pm x\,2^{-2}$$

$$x^{11} \leftarrow x^1 \pm y^1\,2^{-10}$$

$$y^{11} \leftarrow y^1 \mp x^1\,2^{-10}$$

The resulting circuit is shown in Fig 2 7 complete pipeline circuit for the N=16 cascade is shown in Fig 2 8 Thus the VFFT algorithm is more suitable to hardware implementation For hardware implementation of VFFT algorithm, making use of recently evolved CORDIC arithmetic processor chip is novel idea. A monolithic processor developed by Gene L Haviland and A TUSZYNSKI, computers products,quotients and several transcendental functions A bulk of CMOS technology with conservative layout rules is used for the sake of high reliability, low power consumption and good cycle speed The details of this chip may be found in Ref (4)

The details of Winograd algorithm to compute DFT are available in Ref (5).

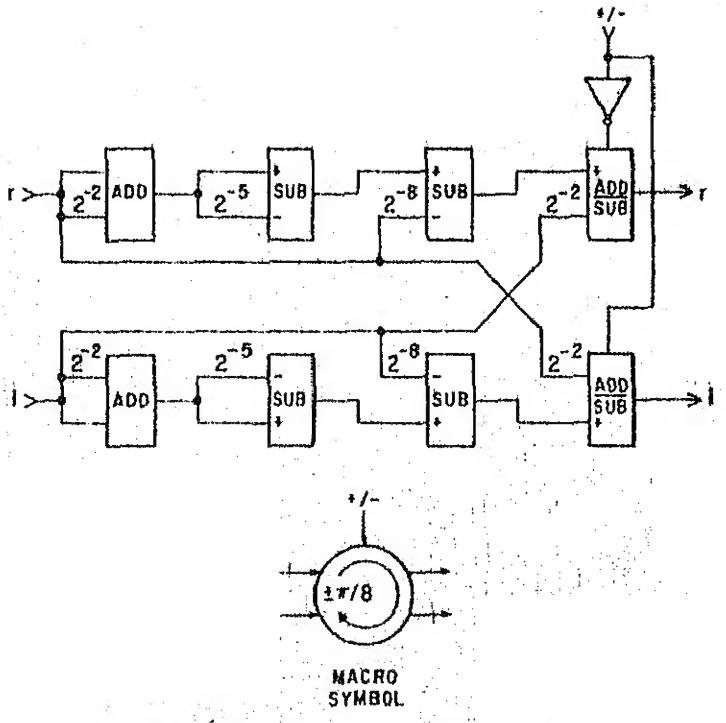


Fig. 2.6 Digital circuit for rotation by $\theta_3 = \pi/8$.

TABLE II

Optimal Vector Rotations, $\Theta_3 = \pi/8$ (22.5°).				
No. of Additions	Error (degrees)	Precision (bits)	Ratio of f_2/f_1	Comments
2	4.0	6.5	(1/2)	
4	.489	9.5	(1/2)(-1/16)	two steps
6	.020	14.0	(12/29)	
8	.00127	18.1	(128/309)	
10	17.8×10^{-6}	24.3	(577/1393)	
12	$.53\times10^{-6}$	30	(2378/5741)	

(a)

Optimal Vector Rotations, $\Theta_4 = \pi/16$ (11.25°).				
No. of Additions	Error (degrees)	Precision (bits)	Ratio of f_2/f_1	Comments
2	2.78	7.0	(1/4)	
4	.060	12.6	(1/5)	
6	.003979	16.5	$(1/5)(-2^{-10})$	two steps
8	9.29×10^{-6}	25.2	(256/1287)	

(b)

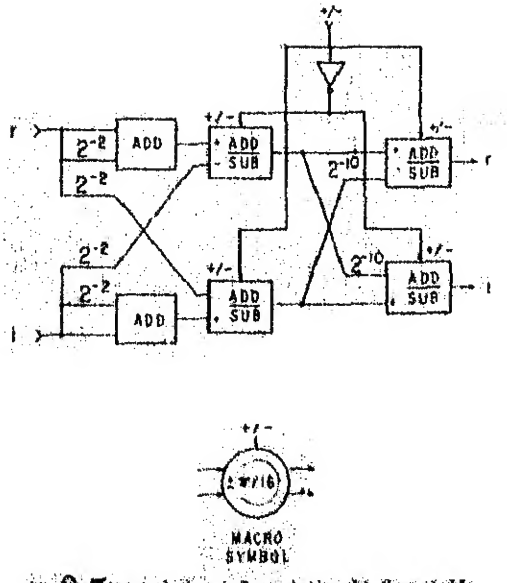


Fig. 2.7 Digital circuit for rotation by $\theta_4 = \pi/16$.

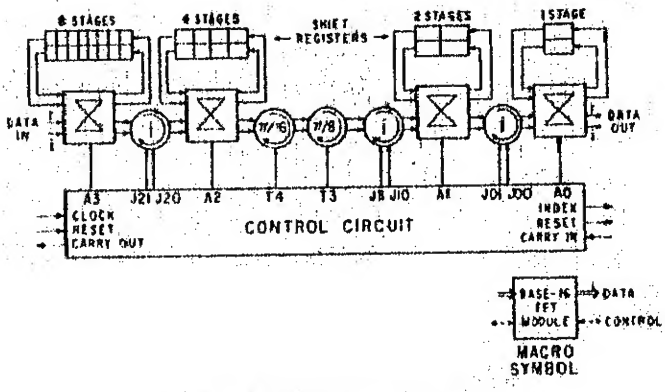


Fig. 2.8 FFT pipeline for $N = 16$.

2 RECURSIVE ALGORITHM

There are many ways of computing DFT recursively The particular method will be selected depending on the application The details of state variable approach to recursively compute DFT can be found in Ref (6)

In some applications it is convenient to segment the complete signal and do the spectral analysis segment wise This is especially so in case of a sophisticated signal f(t) of long duration, for example, a voice record In principle we can uniquely determine f(t) in terms of its transform F(W), however any numerical processing directly involving F(W) is unrealistic because F(W) is too complex to be recovered with sufficient accuracy from its samples To avoid this we propose to describe the spectral properties of f(t) in the form of the running Z-transform or recursive DFT formula

The DFS (Discrete Fourier Series) of N numbers can be obtained from the running Z Transform The following formulation will yield a method to compute DFS in a recursive way

Recursive DFS

Given a sequence f(n) and an integer N, we form the running Z transform

$$\phi(n,Z) = \sum_{k=0}^{N-1} f(n-K)\, Z^{-k}$$

of f(n) For a fixed n, ϕ (n,Z) is the Z transform in the variable K of the segment f(n - k) of f(n) shown in Fig 2 10

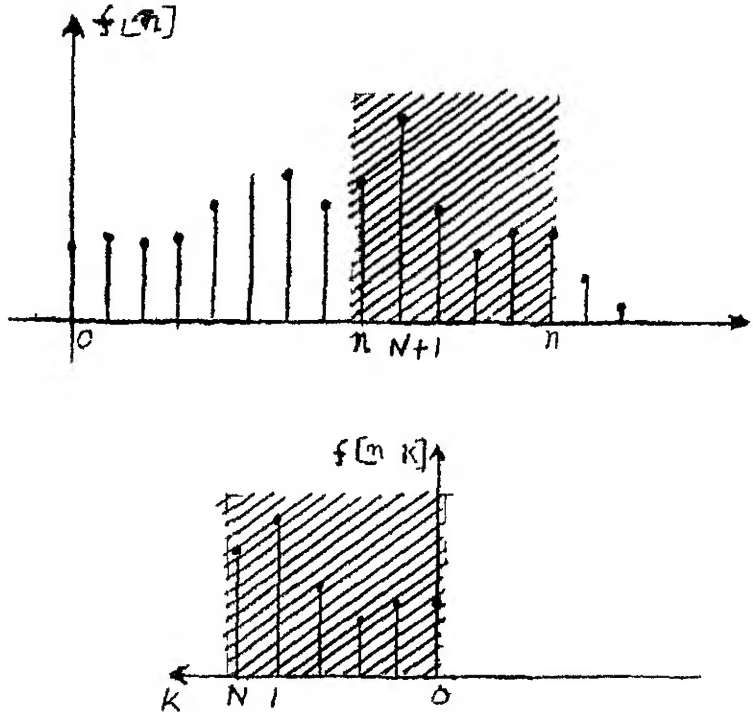


Fig 2.10

The function $\phi(n,Z)$ satisfies the first order recursion equation

$$\phi\ (n,Z) - Z^{-1}\ \phi(n-1, Z) \qquad (2\ 26)$$

$$= f(n) - Z^{-N}\ f(n-N)$$

This can be proved as follows

$$\phi\ (n, Z) = \sum_{k=0}^{N-1} f(n-k)\ Z^{-k}$$

$$\phi(n-1,Z) = \sum_{k=0}^{N-1} f[n-(k+1)]Z^{-k} \quad \text{substituting } k+1 = r$$

$$k = r - 1$$

$$\phi(n-1, z) = \sum_{1}^{N} f(n-r)\, Z^{-(r-1)}$$

$$= Z \sum_{r=1}^{N} f(n-r)\, Z^{-r}$$

$$Z^{-1}\ \phi(r-1,Z) = \sum_{r=1}^{N} f(n-r)\, Z^{-r}$$

$$\phi(n,Z) - Z^{-1}\ \phi\ (n-1, Z)$$

$$= \left[\sum_{k=1}^{N-1} f(n-k)\, Z^{-K} \right] - \left[\sum_{r=1}^{N} f(n-r)\, Z^{-r} \right]$$

$$= f(n) - f(n-N)\, Z^{-N}$$

Therefore, the resulting recursion becomes

$$\phi(n,Z) - Z^{-1}\ \phi(n-1,Z)$$

$$= f(n) - Z^{-N}\ f(n-N)$$

$$\phi\ (n,Z) = \sum_{k=0}^{N-1} f(n-k)\, Z^{-k}$$

if we substitute $Z = e^{j2\pi/N})^{-m} = W^{-m}$

$$\phi\ (n,\ W^{-m}) = \sum_{k=0}^{N-1} f(n-k)\ \ W^{km} \qquad (2\ 27)$$

Equation (2 27) shows that for a fixed n, the DFS of the N numbers f(n-k) equals $\phi\ (n,\ W^{-m})$

The running DFS $\phi\ (n,W^{-m})$ of f(n) satisfies the recursion equation $\phi(n,W^{-m}) - W^{m}\phi(n-1,\ W^{-m}) = f(n)-f(n-N)$ (2 27)

This follows from the previous recursion formulation (eqn 2 26) with $Z = W^{-m}$ and using the fact $W^{N} = 1$

From this recursive equation DFS analyzer is constructed as follows

The recursion equation (2 27) defines a discrete system with input f(n), output $\phi(n,\ W^{-m})$, and system function

$$S(Z,m) = \frac{1 - Z^{-N}}{1 - W^{m}\ Z^{-1}} \qquad (2\ 28)$$

The system is shown in Fig 2.11

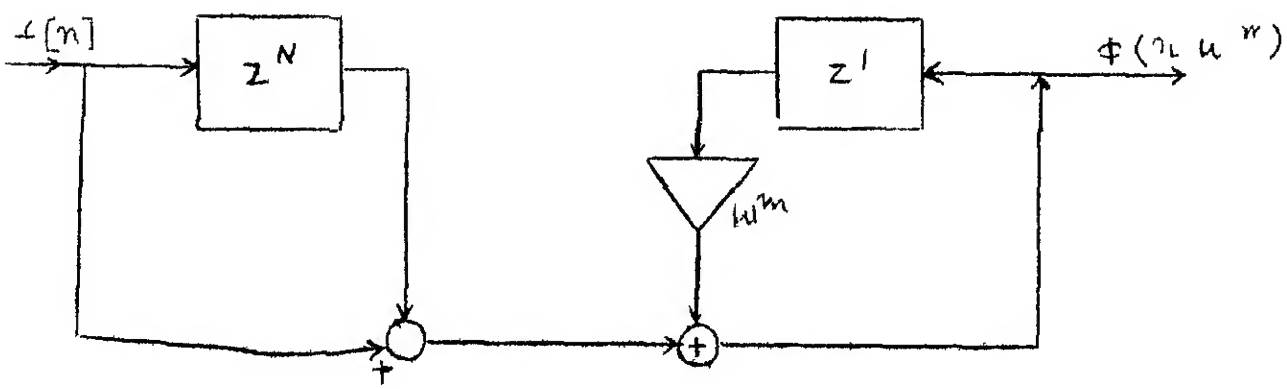
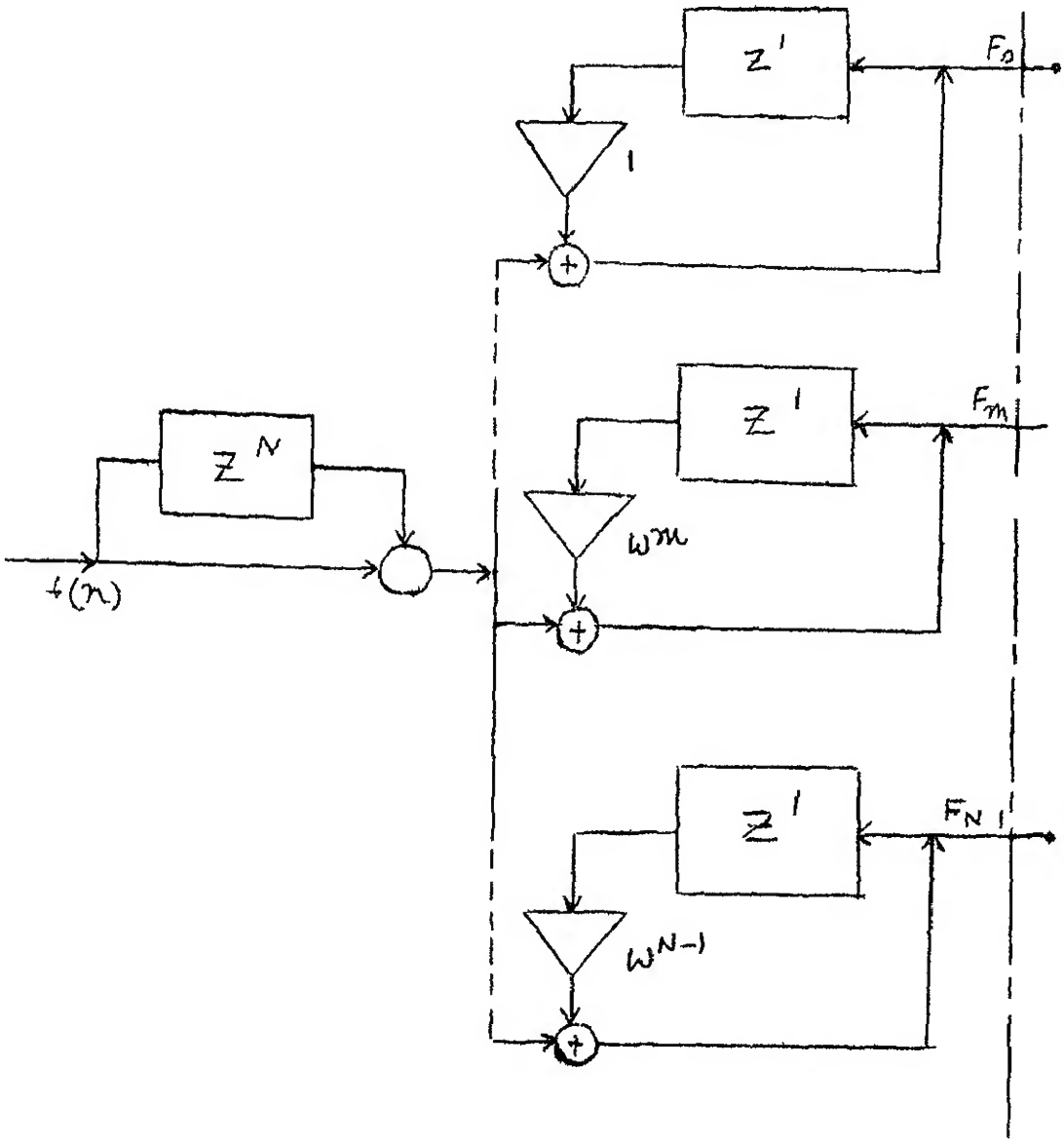


Fig 2 11

The system consists of a shift register with output f(n-N), one delay element and multiplier The shift register can be omitted if we have direct access not only to f(n) but to f(n-N) Connecting N such systems in parallel we obtain the real-time spectrum analyzer shown in Fig 2 12



F_m = DFS Terminals

Fig. 2.12

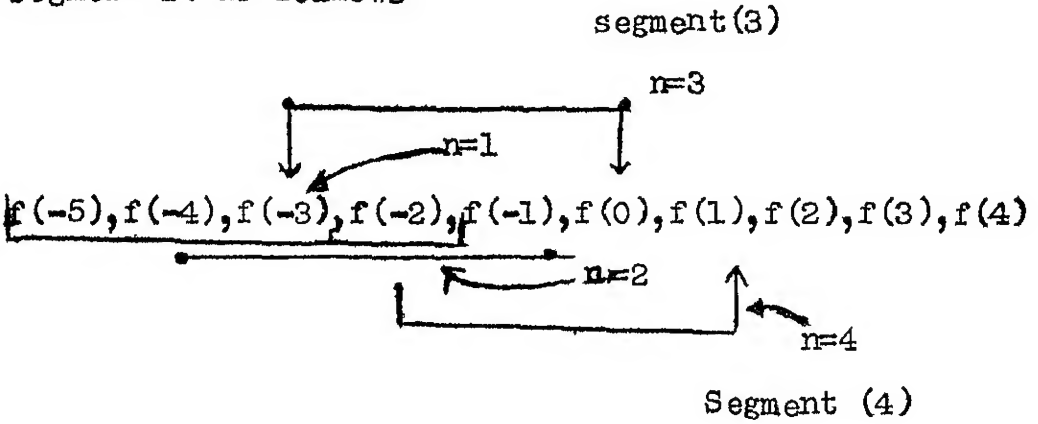
If the input to the analyzer is a sequence f(n), then the outputs at terminals F_m equal the DFS coefficients $\phi(n,W^{-m})$ of the N numbers f(n) f(n-N+1)

Example

Let us say the sequence for which DFS is required to be computed is

f(-5), f(-4), f(-3), f(-2), f(-1), f(0),
f(1), f(2), f(3), f(4), f(5)

Let us say we want to do the spectral analysis for each segment having 4 points Running Z-transform enables us to segment it as follows



If the DFS of segment (4) is to be computed by making use of recurive formula it can be found from DFS of segment(3) and one new and old samples

As a numerical example consider the sequence

2,	5,	-2,	3,	7,	1,	2,	1,	-5
f(-4),	f(-3),	f(-2),	f(-1),	f(0),	f(1),	f(2),	f(3),	f(4)

N = Segment length = 4

To compute, let us say, $\phi\ (4,\ W^{-m})$ for m = 0, 1, we can illustrate the u e of recursion as follows

$$\phi\ (n,\ W^{-m}) = \sum_{k=0}^{N-1} f(n - k)\ W^{km}$$

$N = 4\ ,\ N-1 = 3 \qquad W = e^{j2\pi/N}$

$$W = e^{j2\pi/N} = e^{j2\pi/4} = e^{j\pi/2} = j$$

$$\phi(4,W^{-0}) = \sum_{k=0}^{3} f(4 - k)\ W^{0}$$

$$= f(4) + f(3) + f(2) + f(1)$$

$$= -5 + 1 + 2 + 1 = -1$$

This is actually the segment using samples f(4), f(3), f(2), f(1)

$\therefore \phi(4,\ W^{-0})$ by definition has got the value -1

Similarly by definition the value of $\phi\ (4,\ W^{-1})$ is -7.

These two values can be computed recursively from $\phi(3,W^{-0})$ and $\phi(3, W^{-1})$ If we calculate these two values we get them as 11 and -j5 respectively (can be calculated from definition of $\phi(2,W^{-0})$ and $\phi(2,W^{-1})$

Recursive formula states that

$$\phi(n,W^{-m}) = W^{m}\phi(n-1, W^{-m}) + f(n) - f(n-N)$$

$\phi(4,W^{-0})$ can be obtained from $\phi(3,W^{-0})$ by putting n = 4, m=0 in the above equation

$$\phi(4,W^{-0}) = W^{0}\phi(3,W^{-0}) + f(4) - f(0)$$
$$= 11 + (-5) - 7$$
$$= -1$$

Similarly $\phi(4,W^{-1}) = W^{1}\phi(3,W^{-1}) + f(4) - f(0)$
$$= j(-j5) + (-5) - 7$$
$$= -7$$

Observations: Here one has to fix the segment length The sequence length for which DFS is computed segment wise, needs to be fixed, if the record of the data before a reference time is not available In practice a sequence of zeros of length M-1 is augmented before the data sequence so that only the first sample is needed to start with

An equivalent recursive formula can be derived directly from the definition of DFT This should be so because of the fact that from Z-Transform one can get DFT with proper

transformation The recursive formula derived directly from the definition of DFT is implemented on Micro computer in this work The advantages of recursive computation of DFT are also enumerated in the next section along with its formulation This recursive way of computing DFT is useful especially in situations wherein data for which the DFT is computed are received sequentially and computation of the DFT can not be completed until the final datum is received This is especially suitable if there is sufficient time gap (computational time) between samples By computing the DFT recursively, only a small number of operations need be made after the receipt of the final datum

In the following section the recursive formula is derived. This particular form is used in doing the microprocessor based spectral analysis in this work In the next section advantages of this algorithm when compared with FFT are enumerated

2 8 Derivation of Recursive Formula: As we know for a sequence $Z_0, Z_1, \quad Z_{M-1}$, of sampled waveform data (either real or complex), the DFT is defined as

$$W_j = \sum_{k=0}^{M-1} Z_k \exp(-2\pi ijk/M) \qquad (2\ 29)$$

$$i = \sqrt{-1}$$

for j = 0,1, M-1 An assumption implicit in the definition of W_j, when applied to sampled waveform data, is that the samples Z_0, Z_1 Z_{M-1}, are obtained at equally spaced time intervals T_0

The Moving-window Discrete Fourier Transform (MWDFT)

Let Z_0, Z_1 , Z_m be an infinite sequence of sampled waveform data, where Z_k is the (possibly complex) value of the signal Z(t) obtained at time t = K T_0 Suppose that for n = 0, the DFT of Z_n, Z_{N+1}, Z_{n+M-1} is to be computed, this sequence of DFT's is referred to as the moving-window DFT (MWDFT)

For n = 0, 1, MWDFT can be defined as W_{jn} as

$$W_{jn} = \sum_{k=0}^{M-1} Z_{k+n} \exp(-2\pi ijk/M)$$

$$i = \sqrt{-1}$$

for j = 0,1, M-1

$$W_{j,\,n+1} = \sum_{k=0}^{M-1} Z_{k+n+1} \exp(-2\pi ijk/M)$$

Substituting K+1 = K^1

K = K^1-1

$$W_{j,n+1} = \sum_{k^1=1}^{M} Z_{K^1+n} \exp \frac{-2\pi ij}{M} (K^1-1)$$

$$= \exp\left(\frac{2\pi ij}{M}\right) \sum_{K^1=1}^{M} Z_{K^1+n} \exp\left(\frac{-2\pi ijk^1}{M}\right)$$

$$= e^{\frac{2\pi ij}{M}} \left[\sum_{K^1=1}^{M} Z_{k^1+n} \exp\left(\frac{-2\pi ijk^1}{M}\right) \right] + Z(n) - Z(n)$$

$$= e^{\frac{2\pi ij}{M}} \left[\sum_{k^1=0}^{M-1} Z_{K^1+n} \exp\left(\frac{-2\ ijk^1}{M}\right) - Z(n) + Z_{M+n} \right]$$

Therefore the desired recursive formula is

$$W_{j,n+1} = \left[W_{jn} + Z_{M+n} - Z(n) \right] \exp\left(\frac{2\pi ij}{M}\right)$$

$$W_{j0} = W_j$$

for $j = 0,1, \quad M-1$, where W_j is as defined in equation (2 29)

It can be seen that for each n, the calculation of $(Z_{M+n} - Z_n)$ is independent of the index j of the Fourier coefficients Thus, the total number of operations (an operation is defined as the performance of a complex additon followed by complex multiplication) for each n is just M, neglecting the one-time calculation of $Z_{M+n} - Z_n$ The required computational

speed is therefore such that the M operations must be accomplished in T_0 seconds for a real-time application Thus we have to choose highest frequency of the signal appropriately keeping in view of the computation speed of the particular processor (up Intel 8080 in the present work) at hand In some situations not all of the Fourier coefficients are of interest, in such cases, the total number of operations to be performed is decreased and time per operation can be increased accordingly

If one looks at the way in which DFT is computed by making use of recursive algorithm the following advantages can be attributed to this type of computation

Advantages We can process the data as they come We need not wait until the final sample comes By computing the DFT recursively, only a small number of operations need be done after the receipt of the final datum Even if the recursive method of computing the DFT requires many more operations than would the FFT, the required time measured from when Z_0 is obtained, might be insignificantly different (That is to say that the number of operations needed to compute the DFT by some method is not an accurate measure of that method's computational efficiency) Here, in this method we are saving time as there is no question of waiting for the arrival of last

sample and we are processing the data as they came in sequence If all the samples are stored as in the case of FFT the time required to retrieve the samples from memory area should also be included in the total time computation to have a realistic idea But in recursive way of computation there is no question of storage of samples and accordingly there is no such difficulty of frequent memory references Even the memory area required is small when compared with that required in FFT All these statements are supported by quantitative results in Chapter 5

An added advantage of this method is that the total number of points need not be a power of 2

2 9 COMPUTATIONAL PROCEDURE - AN OUTLINE

MWDFT is defined as

$$W_{jn} = \sum_{k=0}^{M-1} Z_{k+n} \exp\left(\frac{-2\pi i jk}{M}\right)$$

$$j = 0, 1, \quad M - 1$$

MWDFT for n equals zero is nothing but ordinary DFT

$$W_{j0} = W_j$$

The recursive formula states that

$$W_{j,n+1} = [W_{jn} + Z_{M+n} - Z_n] \exp\left(\frac{2\pi i j}{M}\right)$$

$$i = \sqrt{-1}$$

$W_{j0} = W_j$ can be computed from the equation

$$W_j = \sum_{k=0}^{M-1} Z_k \exp\left(\frac{-2\pi ijk}{M}\right) \quad j=0,1, \quad M-1$$

by simply starting with M-1 zeros proceding Z_0, the first sample in the data sequence The computational procedure is described below for M = 4 (4 point DFT)

Let the data sequence be $Z_0^1, Z_1^1, Z_2^1, Z_3^1$ start with 3(=M-1) zeros preceding Z_0^1 So the new sequence may be designated as

0, 0, 0, Z_3, Z_4, Z_5, Z_6

(Z_0, Z_1 Z_2)

$Z_3, Z_4, \ldots Z_{2M-2}$ correspond to the actual data sequece $Z_0^1, Z_1^1, \ldots Z_{M-1}^1$

First we compute $W_{j0} = W_j$ for j = 0,1, M-1(=3) by making use of the formula

$$W_{j0} = W_j = \sum_{k=0}^{3} Z_k \, e^{\frac{-i2\pi}{M} jk}$$

It can be observed that while computing W_{j0} for j=0, . 3 (=M-1) only Z_{M-1} (Z_3) has got non zero value. So the simplification in computation at this step result This will give

W_{00}

W_{100}

$W_{M-1,\ 0}$

Then taking the recursive formula

$$W_{j,\ n+1} = \{ W_{jn} + Z_{M+n} - Z_n \} e^{i \frac{2\pi j}{M}}$$

and , first n=0 is substituted and get W_{j1} for j=0, M-1 by making use of W_{j0} for j=0, M-1 and the next sample in the sequence i e , Z_{M+n} (= Z_{4+0}) Finally when n=2 (n=M-2) we get final set of coefficients from $W_{j,3}$ which makes use of W_{j2} and Z_{2M-2} the last datum The final set of coefficients which we compute after the arrival of last datum are exactly equal to the coefficients that are obtained by making use of the definition Recursive algorithm is implemented on MICRO-78 computer and results are obtained

CHAPTER 3

MICROPROCESSOR IMPLEMENTATION DETAILS

A recent and fascinating innovation in the digital electronics field is the development of the microprocessor using large-scale integrated technology The microprocessor, combined with the memory and input/output devices, forms a micro-computer whose cost is competitive with conventional random logic in an ever-increasing number of applications The micro-computer is not only finding use in small systems, where it may replace a mini-computer, but it is also opening vast new areas of applications where larger machines are not economically feasible

The microprocessor can be used for real-time applications and dedicated systems can be developed making use of single board computer (SBC) and related transducers With SBC and transducers, A/D and D/A converters and support-hardware the spectrum analysis can be done in real time This type of spectrum analyzer will be very cheap to realize using microprocessors, especially with the advent of fast and cheap processors which are easily available in the market This realization will have the advantages of low cost and high reliability when compared with spectrum analyzers realized with

combinational and sequential logic circuitry

In this Chapter the details of implementation of the recursive algorithm on micro-computer are given The assembly language listing of th program can be found in Appendix A

Performance limitations (spe d,shorter word length, limited addressing modes, fewer internal registers) limit the competitiveness of micro-computers So while selecting a microprocessor for a particular application we have to make suitability study of the processor at hand, and then only one particular type of processor should be selected from the few which are readily available

3 1 CHOICE OF THE MICROPROCESSOR

Motorola 6800, Intel 8080 and Fairchild F-8 are the few readily available processors For this project work Intel 8080 processor is selected The choice of Intel 8080 was based on the standard support software available with it An Intel 8080 based ECIL'S micro-computer MICRO-78 with the facility of a floppy disk, resident assembler and also with an efficient debugging aid (on-line Debugging system) is chosen for the DFT algorithm implementation Description of the MICRO-78 and Intel 8080 processor for our application is given in Ref [11]

In this computer the data word length is 8 bits But to represent numbers with reasonable degree of accuracy it is required to have 16 bits This is so because 8 bits are to be spared for fractional part and another 8 bits for integer part of the number So two registers (Register pair) are used while doing arithmetic operations

3 2 IMPLEMENTATION DETAILS

The DFT computation requires values of sinusoidal and Cosinusoidal function values These can be generated by making use of trigonometric function routines provided in Decimal Arithmetic Packages (DAP) in MICRO 78 But this is not preferred because the computation of trigonometric values by making use of the above said routines will reduce the computational speed of the processor drastically. It also increases the storage requirement Instead of all these values, depending on the number of points for which the DFT is computed are stored in the form of Look up Tables in certain blocks of MICRO-78 memory area These values are made use of while computing DFT The placing of these valued in the overall memory allocation is shown in the block diagram of memory structure for the recursive algorithm in Fig. 3.1

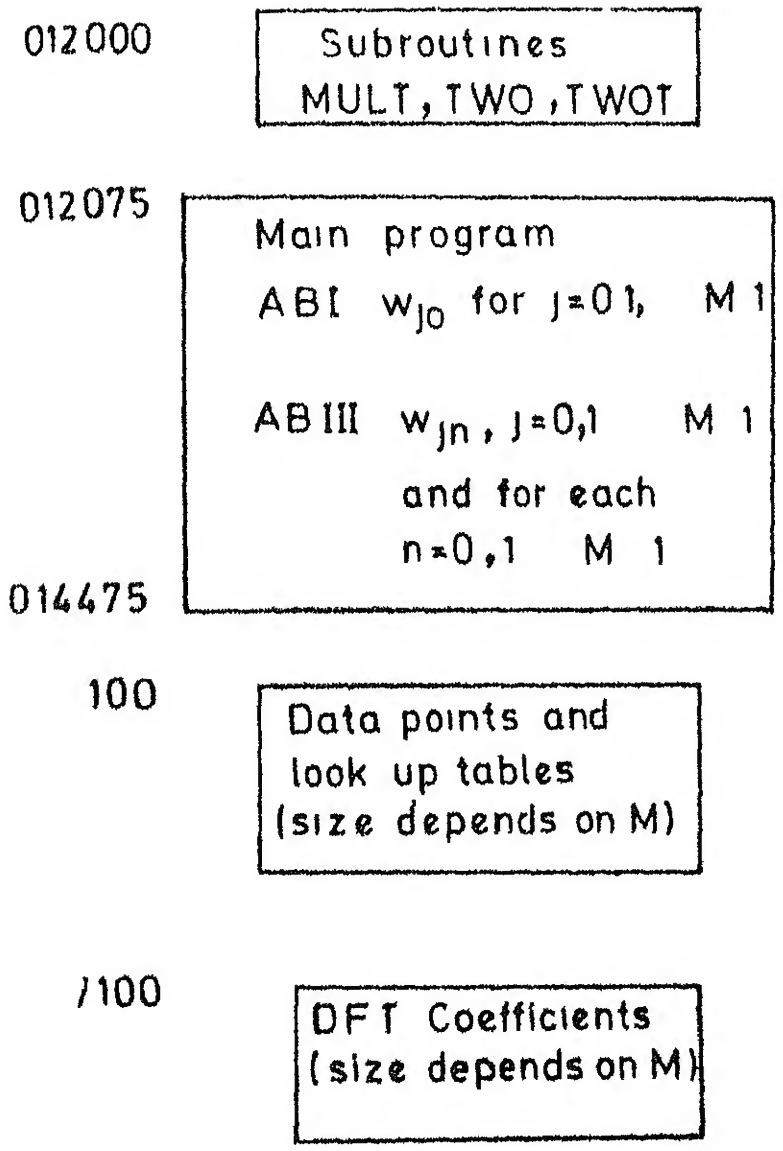


FIG 31 RECURSIVE DFT PROGRAM MEMORY STRUCTURE

3 3 ROUTINES USED

MULT This is a 16 x 16 multiplication routine While writing this program every care has been taken to see that the total time for 16 x 16 multiplication is the least Multiplicant is 8 bit integer and 8 bit fractional part Similarly 16 bits are allocated for multiplier The result is 32 bits 2 bytes integer and 2 bytes fraction

But it is truncated to 1 byte integer and 1 byte fraction for simplicity as it is not going to affect the accuracy much This routine is called in the main program as and when the need arises

TWO: This is a routine which will compute the two's complement of a 16 bit number This sub-routine is made use of in the subtraction of numbers

TWOT : This is also a routine which is useful in the subtraction of two 16 bit numbers This will introduce the sign of the resultant number and stores it somewhere for further use

3 4 PROCEDURE FOR COMPUTATION OF FREQUENCY COMPONENTS

Here the procedure which is followed to compute Fourier Components is outlined In general the frequency components may be complex The use of complex exponential routine already

available in DAP routines of MICRO-78 is eliminated for the same reasons as outlined earlier (computational speed aspect) The complex Fourier components are spilt into real and imaginary and they are computed seperately This will reduce the overall computational time and enable th program to work for real time applications

The DFT is defined as

$$W_j = \sum_{k=0}^{M-1} Z_K \exp\left(\frac{-2\pi ijk}{M}\right) \qquad (3.1)$$

$$i = \sqrt{-1}$$

$j = 0,1, \quad M-1$

Wj_n the MWDFT is defined as

$$Wj_n = \sum_{k=0}^{M-1} Z_{k+n} \exp\left(-2\pi ijk/M\right) \qquad (3\ 2)$$

$j = 0,1, \quad M-1$

The final recursive formula says that

$$W_{j,\ n+1} = \left[W_{jn} + Z_{M+n} - Z_n \right] e^{\frac{2\pi ij}{M}} \qquad (3.3)$$

$j = 0,1, \quad M-1 \qquad i = \sqrt{-1}$

Step 1

Computation of W_{jn} for $n = 0$ and $j = 0,1, \quad M-1$, we get a sequence $W_{00}, W_{10}, W_{20} \quad W_{M-1}, 0$ in this step

Each number is a complex number They are computed as follows

$$W_{j0} = W_j$$

As $$W_j = \sum_{k=0}^{M-1} Z_K \exp(-2\pi i jk/M)$$

$$j = 0, 1, \quad M-1$$

and because of the fact Z_0, Z_{M-2} becomes zeros when we start with M-1 zeros in the beginning of the data sequence,

$$R(W_j) = W_j \text{ REAL PART} = Z_{M-1} X[\cos \frac{-2\pi}{M} xj \; x(M-1)] \quad (3\,4)$$

$I_m(W_j)$ = Imaginary Part of W_j

$$= Z_{M-1} X \sin [\frac{-2\pi}{M} xjx(M-1)] \quad (3\,5)$$

$$W_j = R(W_j) + j \; I_m(W_j) \quad (3\,6)$$

The assembly language program starting with lable ABI and memory location 012164 will compute W_{j0} for j = 0,1, M-1 (both Real and imaginary components) W_{j1}, W_{j2} $W_{j,M-2}$, W_j, M-1 are computed from the recursive formula for j=0,1, M-1

$$W_{j,\,n+1} = [W_{jn} + Z_{M+n} - Z_n] \; e^{\left(i \frac{2\pi j}{M}\right)}$$

$$\text{REAL}(W_j,\ n+1) = [\text{Real}[W_{jn}] + Z_{M+n} - Z_n\ ]\ \text{COS}\left(\frac{2\pi}{M} j\right)$$

$$- (\text{Imaginary } W_{jn})\text{SIN}\left(\frac{2\pi}{M} j\right) \qquad (3\ 7)$$

$$\text{IMAG}(W_j\ ,\ N+1) = \text{IMAG}(W_{jn})\text{COS}\ \frac{2\pi}{M} j\ + [\text{REAL}(W_{jn}) + Z_{M+n} - Z_n\ ] \times$$

$$\text{SIN}\left(\frac{2\pi}{M} j\right) \qquad (3\ 8)$$

The samples are assumed to be real in the implementation of the algorithm on MICRO-78

REAL (W_j, n+1) and IMAG (W_j, n+1) are computed accordingly for n=0 This gives W_{j1} values for j=0, 1 , M-1 Similarly making use of equations (3 7) and (3 8) W_{j2}, W_{j3} $W_{j,M-1}$ are computed by substituting n=1,2, M-2 The final DFT coefficients are obtained when n=M-2 is substituted (W_j, M-1 values for j= 0,1, M-1)

The assembly program listing starting from memory location 012471 and label AB111 gives these componets.

Thus the final set of components $W_{0,M-1}$, $W_{1,M-1}$, $W_{2,M-1}$, $M_{M-1,M-1}$ will give the DFT of samples Z_0, Z_1 , Z_{M-1}

CHAPTER 4

E X A M P L E S

This Chapter undertakes various examples, which are considered in this work to illustrate the DFT computation of a given set of data points making use of recursive algorithm The succeeding section, discusses briefly the selection considerations of various parameters in the DFT computation

4 1 SELECTION OF DFT PARAMETERS

It is assumed that the discrete-time signal is derived from a continous time function which is a representation of a signal The major DFT parameters are as follows

T = Increment between time samples (in seconds)

f_s = sampling rate (in Hertz) = 1/T

F = increment between frequency components (in Hertz)= frequency resolution

t_p = record length (in seconds)=effective period of the time signal = 1/F

f_o = folding frequency = $f_s/2$ (in hertz)

f_h = highest possible frequency in spectrum (in hertz)

N = number of samples in the record

In order to avoid aliasing, it is necessary that

$$f_s \geq 2 f_h \qquad (4\ 1)$$

This implies that T must be selected according to the relation

$$T \leq 1/2f_h \qquad (4\ 2)$$

The increment between spectral components (F), can be thought of as frequency resolution For a desired frequency resolution the minimum record length t_p must be selected according to

$$t_p \geq 1/F \qquad (4\ 3)$$

The study of equations (4 2) and (4 3) leads to the conclusion that there is a trade off between the bandwidth of the system and the frequency resolution between successive samples To be able to handle signals of larger bandwidth, it is necessary that T be reduced For a given N, this would shorten the record length and, thus, decrease the frequency resolution Conversely, to increase the resolution, it is necessary to increase t_p For a given N this would increase T, which would restrict the bandwidth of operation

The only way in which either the frequency resolution or the bandwidth of operation can be increased while holding the other constant is to increase the number of points N in a record length If f_h and F are both specified, then, N must satisfy

$$N \geq 2f_h/F \qquad (4\ 4)$$

In all the examples considered in this work the number of points for which DFT is computed is selected based on the above discussion

4 2 VARIOUS EXAMPLES FOR DFT COMPUTATION

Example 1

100-point DFT analysis to a pure sinusoid of frequency 500 Hz and maximum amplitude of 5 and 10 Hz frequency resolution The DFT spectrum (plot of |X(m)| VS m) is shown in Fig 4 1 A peak, as desired, at the frequency of the sinusoid is observed.

Example 2

Spectral analysis of a rectangular window shown in Fig 4 2

Here 128 points DFT is computed and the sp ctrum is plotted It is clear from the figure that the amplitude of the frequency components decrease as frequency increases and the envelope has the usual theoretical form of SIN(x)/x The plot of |X(m)| VS m is shown in Fig 4 3

Example 3

Power spectrum estimate of a rectangular pulse shown in Fig 4.4 is obtained after computing the 9-point DFT by recursive algorithm The envelope of the power spectrum estimate is

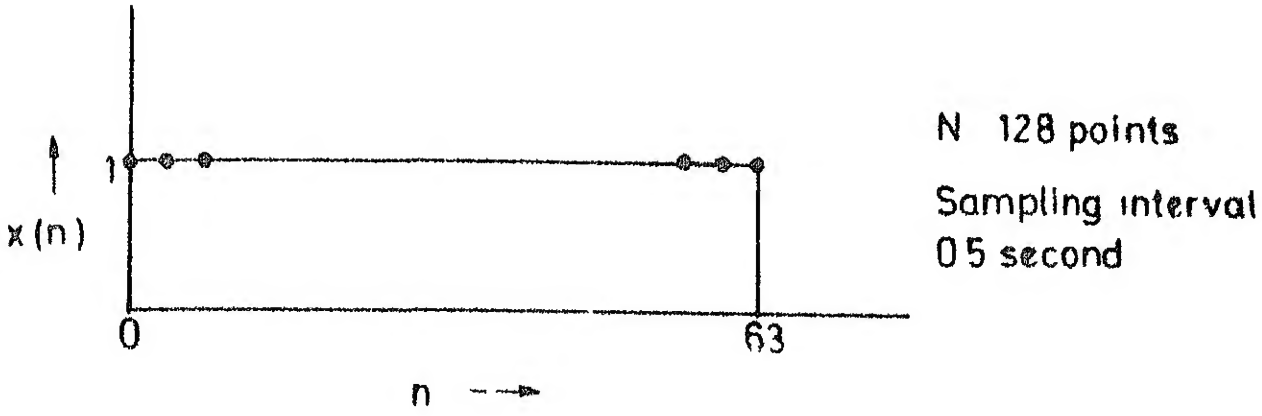


FIG 4 2 RECTANGULAR WINDOW FOR SPECTRAL ANALYSIS

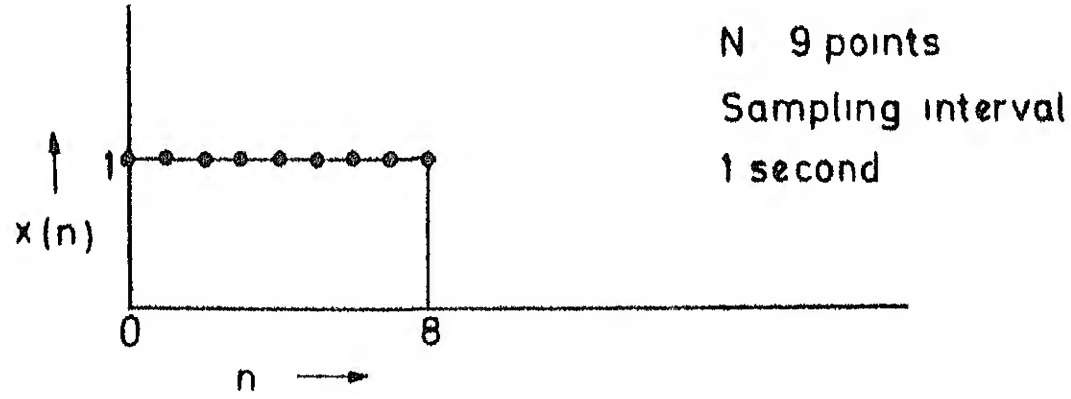


FIG 4 4 RECTANGULAR WINDOW FOR POWER SPECTRUM ESTIMATION

shown in Fig 4 5 The periodogram estimate of the PSD is computed from following the formula,

$$S_N(W) = \frac{1}{N} | X_N(W) |^2$$

where $X_N(W)$ is the DFT of N Points

Example 4

By passing a Pseduo White-Guassian noise sequence of uniform power spectral density through a low pass digital filter which is obtained from the analog transfer function of the Butterworth filter, the magnitade square characteristic of the low pass filter is obtained The Butterworth filter characteristic chosen is

$$G(P) = \frac{1}{1 + 1\ 4\ 142136P + P^2}$$

The 3dB cut off frequency of the filter is 50Hz The sampling rate of the system is 500 Hz The magnitude square characteristic can be found in Fig 4 6 The variations in the power spectrum of pseudo-random noise sequence generated on the computer are reflected as variations in the filter characteristic obtained

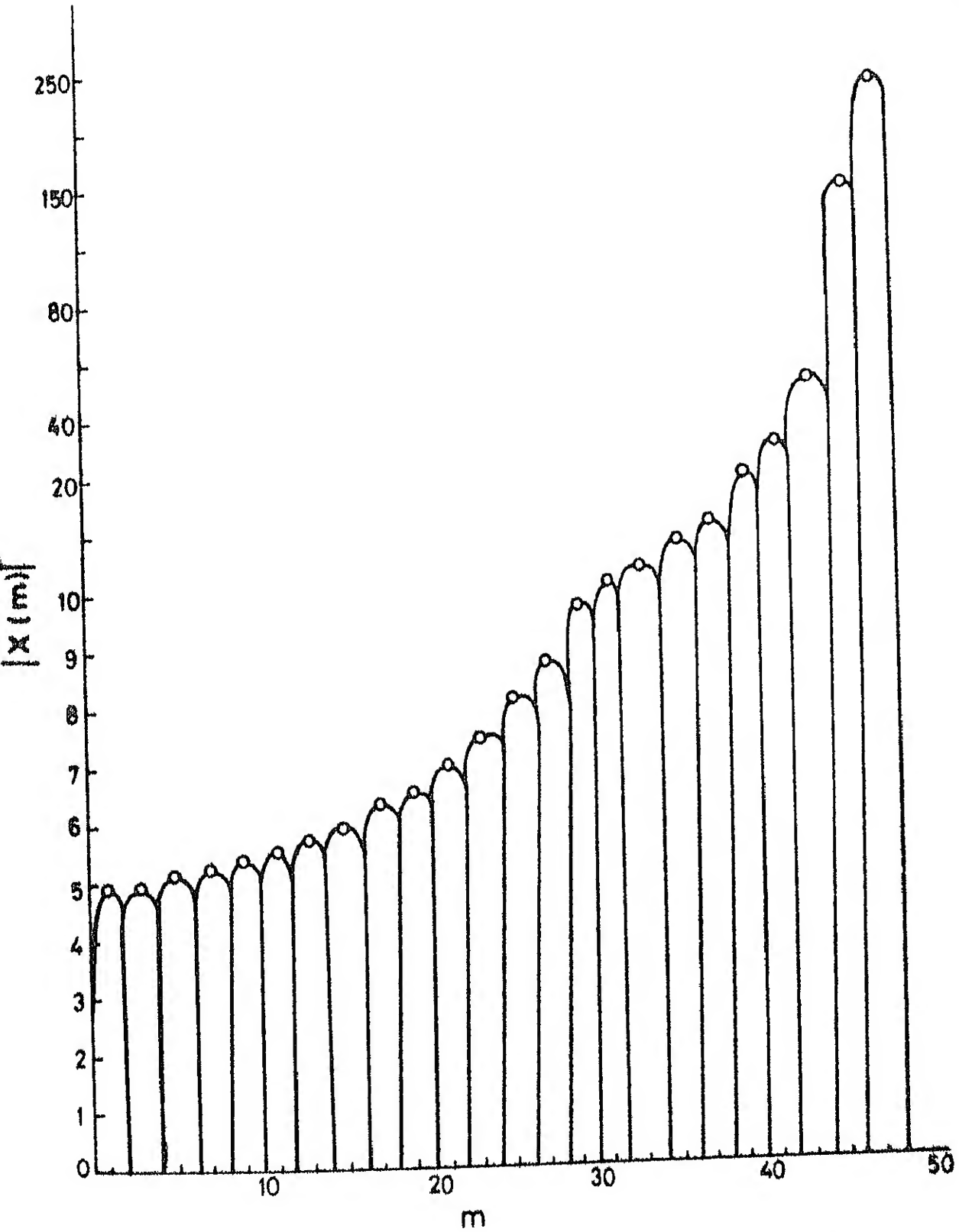


FIG 4.1 DFT ENVELOPE OF A PURE SINUSOIDAL SIGNAL (HALF CYCLE)

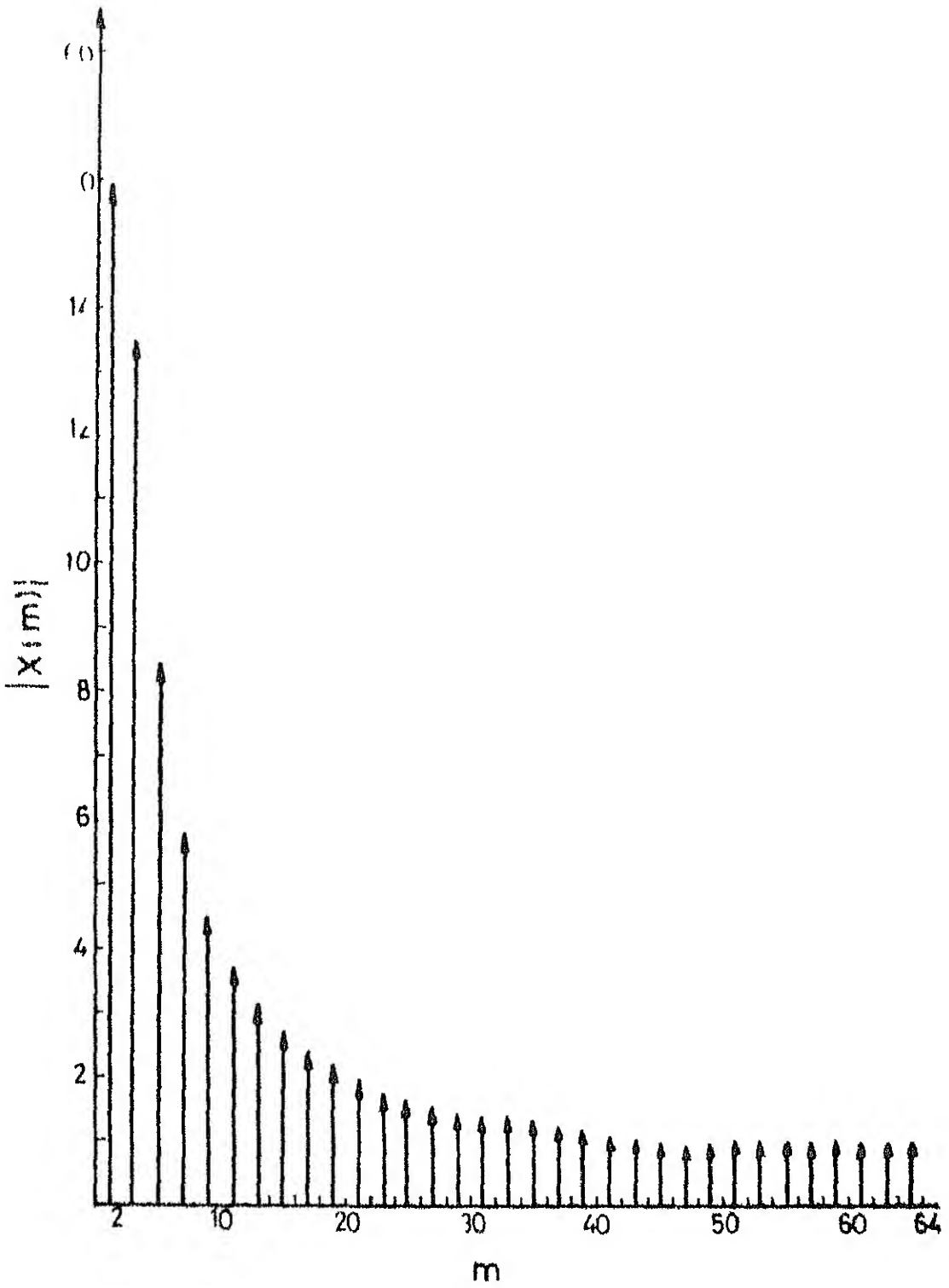


FIG 4.3 DFT SPECTRUM OF A RECTANGULAR WINDOW (HALF CYCLE)

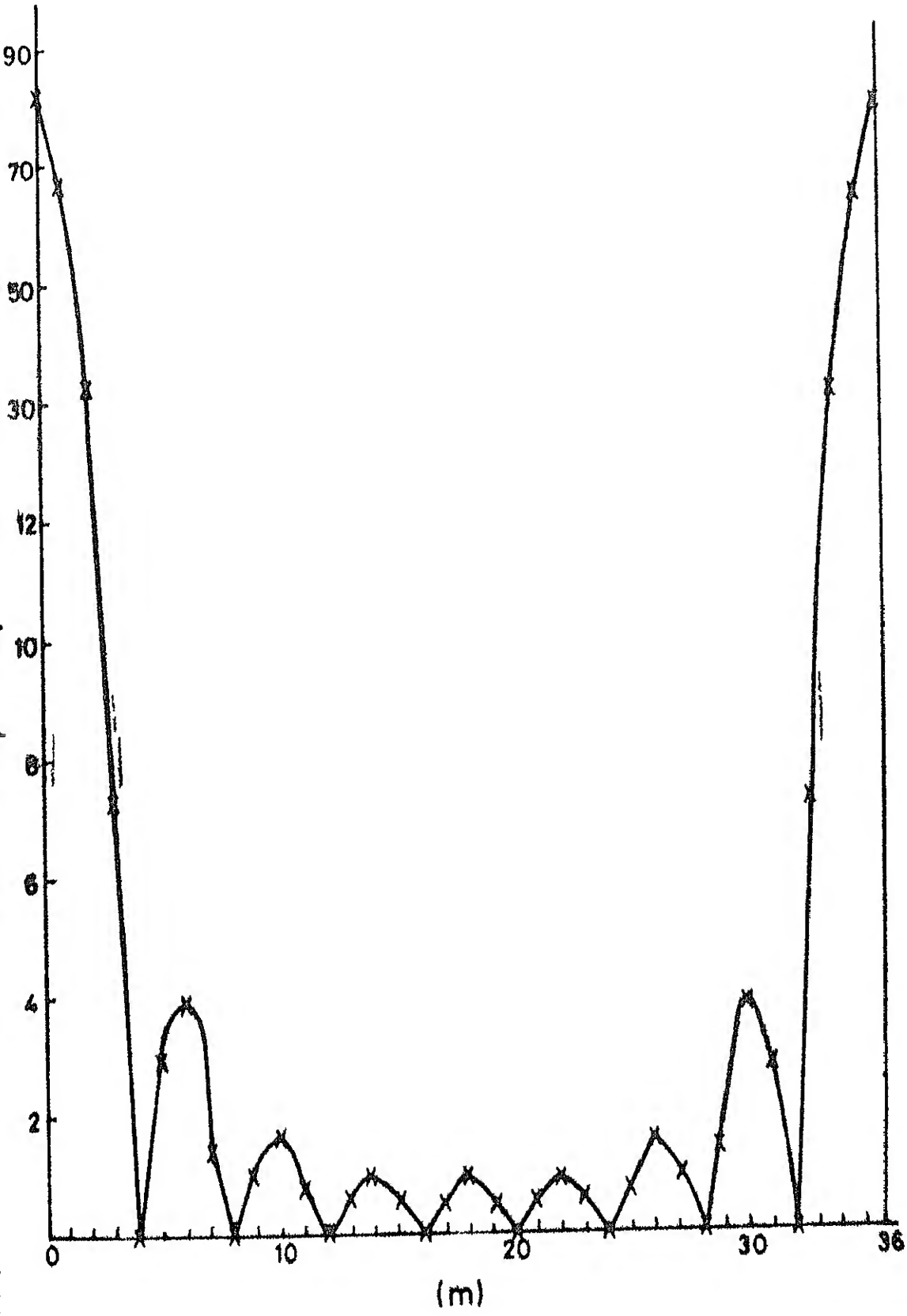


FIG 4.5 POWER SPECTRUM ENVELOPE OF A RECTANGULAR WINDOW

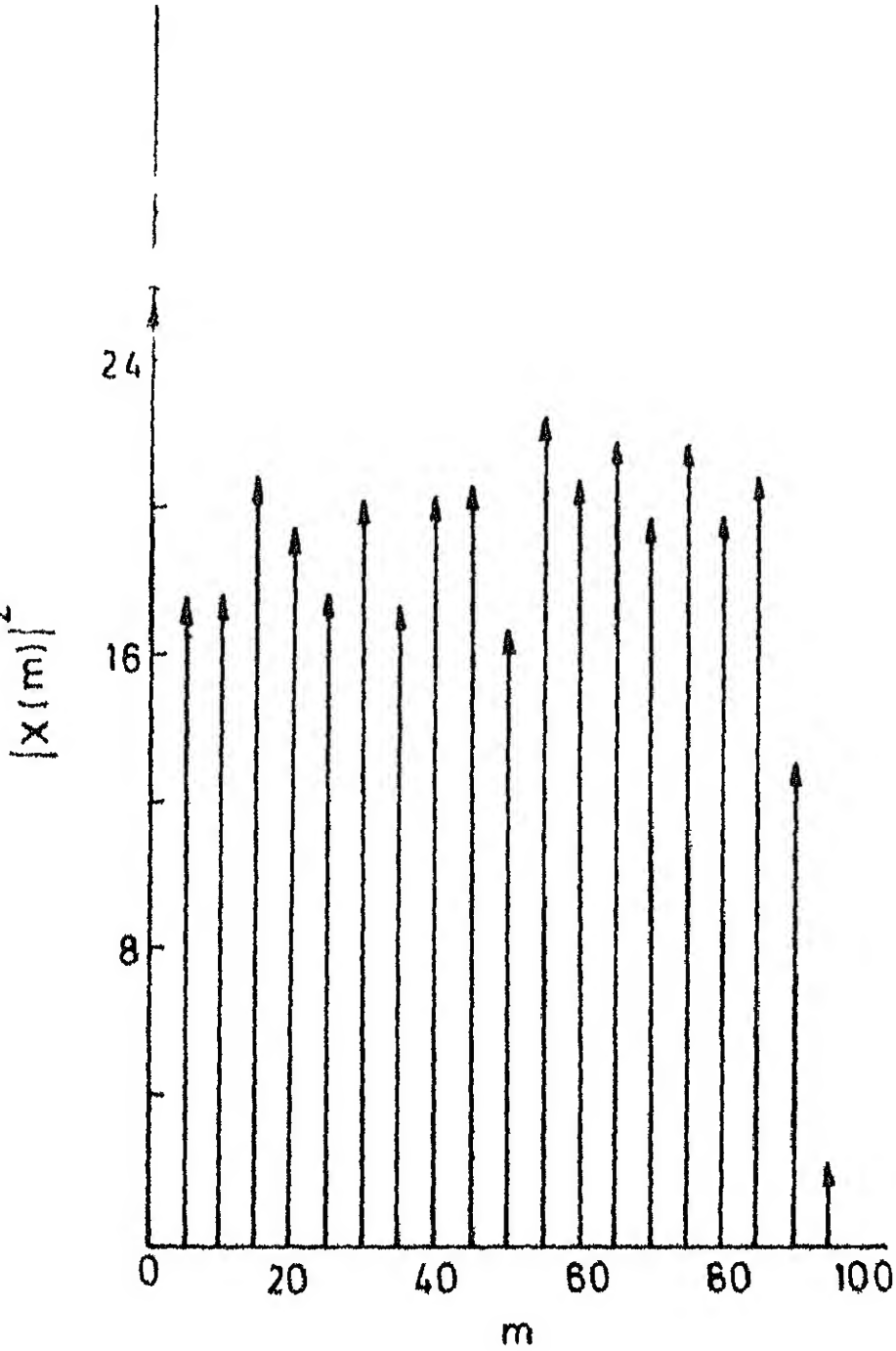


FIG 4.6 MAGNITUDE SQUARE CHARACTERISTIC FOR EXAMPLE 4

CHAPTER 5

COMPARATIVE STUDY

The comparative study of the two algorithms to compute DFT of a set of given data points namely 1) FFT 2) Recursive algorithm can be done at two fronts 1) Computational time required to perform DFT 2) storage requirements

5 1 COMPARISON OF COMPUTATIONAL TIME

While comparing computational time required by the two algorithms it should first observed that FFT algorithm gives DFT for a set of given sample points where as recursive way of computing DFT takes sample points one by one and displays the DFT coefficients computed for data upto that instant To be specific, FFT algorithm has to perform $M/2 \log_2 M$ complex multiplications to get the M -DFT coefficients where as M^2 operations are required in DFT by recursive algorithm But to be realistic, one has to start counting time from the arrival of the first sample uptil the arrival of last sample for computational time calculation This is nothing but the signal acquisition time This depends on the particular sampling rate used

In this work a logic state analyzer (Type HP 1610) has been connected to MICRO-78 computer and the computational time required to compute all the coefficients at a particular

instant of time is obtained The timings for different sizes of data are listed in the following table

Table 5 1

No of Points (Complex)	Computational time
100	677 m sec
128	722 m sec
256	1415 m sec

To get the final DFT coefficients these times have to multiplied by the data size (acually less than this is observed), because in the program one particular loop has to be computed recursively M times For M=128 the computational time is found to be 7225 m Sec , or 7 Sec , 225 m sec If we take the computational time required to perform the same operation using FFT, it is found from Ref (10), to be equal to u sec 500 m sec This is just computational time If one adds the acquisition time which has to be added for comparison purposes, the time required for computing DFT by recursive algorithm may be insignificantly different from that of FFT The term 'insignificantly different' is used because one can compare the two methods only for certain sampling rates of the signal Especially if the sampling interval is more and the data size for which

algorithm also 'in-place' notion is employed the saving in storage area is obtained When we compare FFT in-place algorithm to recursive algorithm there is no improvement in memory size But in the FFT algorithm computational speed decreases So one has to select particular type of algorithm depending on practical constraint

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

Concluding Remarks

Assembly language program for Intel 8080 microprocessor based computer (ECIL'S MICRO-78) for the purpose of implementing recursive algorithm for the computation of DFT has been developed in present work This is useful in carrying out spectral analysis for real-time applications Same algorithm is also implemented on DEC-10 computer and results are obtained

This type of computation is suitable when the data are coming sequentially and the sampling interval is sufficient enough to perform M operations for an M-point DFT Here, as DFT is computed from the definition there is no necessity for unscrambling of the output or scrambling of input data as required in FFT algorithm So there is no requirement of bit reversal sub-routines in the computer programe Even if the recursive method of computing the DFT requires more operations than would the FFT, the required time measured from the time when the first sample arrives might be insignificantly different, because of no waiting time and fewer memory references Another advantage here is that M, the number of points,

need not be a power of 2

As the DFT algorithm is implemented in this work on an Intel 8080 microprocessor based micro-computer (ECIL'S MICRO-78) and because of the computational time to perform M operations is of the order of few hundreds of m sec the spectral analysis in real time is restricted to input signals of bandwidth less than few Hz This is due to low clock frequency and 8 bit word size of the processor For representing numbers with reasonable accuracy atleast 16 bits are required, and so double precision has been employed This reduces the computational speed to some extent The 16 x 16 software multiplication routine which is developed and used in this work has taken approximately 1 25 m sec This can be reduced to approximately 350 micro-seconds if one develops hardware multiplier using TRW 1010J (for 16 x 16 bit multiplication) chips. As the number of multiplications mainly determines the total computational time, with the processor used in this work, only low frequency signals may be handled with real time constraint

Some recently evolved algorithms for computing DFT efficiently are also presented This is with a view of the possibility of efficient and economic realizations of spectrum analyzers in future.

Future Work :

There is scope for further work based on the recursive way of computing DFT This is mainly so because of the increasing number of real-tim applications One can actually build such a real-time spectrum analyzer by burning the devoloped softwaro progran, for th recursive implementation of the DFT, into an EPROM (erasable programable read-only-memory, chips like 1702A) and using it in the realization In this, software multiplication routine can be replaced by a hardwaro multiplior built using TRW 1010 J chip

Using this typo of 16 x 16 bit hardware multiplier and single board computer (SBC) kits having fast processors like Intel 8086 or Zilog 8000, appropriate interfacing hardware and a display unit, one can build an economic and highly reliable real-timo spectrum analyzer

An attempt can be made to extend the recursion concept of for DFT computation to compute other orthogonal transforms like Walsh-Hadamard transform (WHT) There may be some advantages in terms of storage requirements in a computer if WHT is implemented recursively As the basis set in WHT involves only -1 and +1 there may not be an increase in computational time because of the increased number of opera-

tions in the recursive way of computing WHT In this type of computations one should strike a compromise between memory size and computational speed The economy in terms of memory storage and recursion in memory references in computing WHT by recursion may even override the loss in computational speed This is an exercise worth trying

REFERENCES

1 BRIGHAM, E O , 'The Fast Fourier Transform', Prentice Hall, Inc , 1974

2 VOLDER, J E , 'The CORDIC Trigonometric Computing Technique' IRE Transactions on Electron Comput , Vol EC-8, pp 330-334, Sept 1959

3 DESPAIN, A.M , 'Very Fast Fourier Transform Algorithms Hardware for Implementation, IEEE Transactions on Computers, Vol C-28, No 5, pp 333-341, May 1979

4 HAVILAND, G L , and TUSZYNSKI, A A , 'A CORDIC Arithmetic Processor Chip', IEEE Transactions on Computers, Vol C-29, No 2, pp 68-79, February, 1980

5 McClellan, 'Numbers Theory in Digital Signal Processing', 1979

6 HOSTETTER, G.H , 'Recursive Discrete Fourier Transform', IEEE Transactions on Accoustics, Speech and Signal Processing, Vol ASSP - 28, No 2, pp 184-189, April 1980

7 DILLARD, GEORGE M , 'Recursive Computation of the Discrete Fourier Transform with Applications to an FSK Communication Receiver', National Telecommunications Conference pp NTC 74-263, 1974

8 AHMED, N , RAO, K R , 'Orthogonal Transforms for Digital Signal Processing', Springer-Verlag, Berlin, 1975

9 SCHWARTZ, M , SHAW, L , 'Signal Processing, Discrete Spectral Analysis, Detection and Estimation', McGraw-Hill, Kogakusha, Ltd , 1975

10 PAPOULIS, A , 'Signal Analysis', McGraw-Hill Book Company, N Y , 1977

11 ECIL'S MICRO-78 General Purpose Software Manual

12 DESPAIN, A.M , 'Fourier Transform Computers Using CORDIC Iterations', IEEE Transactions on Computers, Vol C-23, No 10, pp 993-1000, October 1974

13 CHUGH, K L , 'A FFT Program for Micro-computer for Real time Application', M Tech Thesis, August 1978, Department of Electrical Engineering , IIT, Kanpur